

AMENDMENTS TO THE CLAIMS

This listing of claims will replace all prior versions, and listings, of claims in the application:

Listing of Claims:

1 1. (Currently amended) A method for executing a fail instruction to
2 facilitate transactional execution on a processor, comprising:
3 executing a start transactional execution instruction to start transactionally
4 executing a block of instructions within a program;
5 wherein changes made during the transactional execution are not
6 committed to the architectural state of the processor unless the transactional
7 execution successfully completes; and
8 if the fail instruction is encountered during the transactional execution,
9 terminating the transactional execution without committing
10 results of the transactional execution to the architectural state of
11 the processor, ~~wherein terminating the transactional execution~~
12 ~~involves branching to a location specified by the fail instruction~~
13 retrying the transactional execution a specified number of
14 times, and
15 if the fail instruction continues to be encountered, obtaining
16 a lock for the block of instructions.

1 2. (Original) The method of claim 1, wherein terminating the transactional
2 execution involves discarding changes made during the transactional execution.

1 3. (Original) The method of claim 2, wherein discarding changes made
2 during the transactional execution involves:
3 discarding register file changes made during the transactional execution;
4 clearing load marks from cache lines;
5 draining store buffer entries generated during transactional execution; and
6 clearing store marks from cache lines.

1 4. (Original) The method of claim 1, wherein terminating the transactional
2 execution additionally involves branching to a location specified by a
3 corresponding start transactional execution (STE) instruction.

1 5 (Canceled)

1 6. (Original) The method of claim 1, wherein terminating the transactional
2 execution additionally involves attempting to re-execute the block of instructions.

1 7. (Original) The method of claim 1, wherein if the transactional execution
2 of the block of instructions is successfully completes, the method further
3 comprises:
4 atomically committing changes made during the transactional execution;
5 and
6 resuming normal non-transactional execution.

1 8. (Original) The method of claim 1, wherein potentially interfering data
2 accesses from other processes are allowed to proceed during the transactional
3 execution of the block of instructions.

1 9. (Original) The method of claim 1, wherein if an interfering data access
2 from another process is encountered during the transactional execution, the
3 method further comprises:

4 discarding changes made during the transactional execution; and
5 attempting to re-execute the block of instructions.

1 10. (Original) The method of claim 1, wherein the block of instructions to
2 be executed transactionally comprises a critical section.

1 11. (Original) The method of claim 1, wherein the fail instruction is a
2 native machine code instruction of the processor.

1 12. (Original) The method of claim 1, wherein the fail instruction is
2 defined in a platform-independent programming language.

1 13. (Currently amended) A computer system that supports a fail instruction
2 to facilitate transactional execution, comprising:

3 a processor; and

4 an execution mechanism within the processor;

5 wherein the execution mechanism is configured to execute a start
6 transactional execution instruction to transactionally execute a block of
7 instructions within a program;

8 wherein changes made during the transactional execution are not
9 committed to the architectural state of the processor unless the transactional
10 execution successfully completes; and

11 wherein if the fail instruction is encountered during the transactional
12 execution, the execution mechanism is configured to;

13 | terminate the transactional execution without committing
14 | results of the transactional execution to the architectural state of
15 | the processor, ~~wherein terminating the transactional execution~~
16 | ~~involves branching to a location specified by the fail instruction~~
17 | retry the transactional execution a specified number of
18 | times, and
19 | if the fail instruction continues to be encountered, obtain a
20 | lock for the block of instructions.

1 14. (Original) The computer system of claim 13, wherein while
2 terminating the transactional execution, the execution mechanism is configured to
3 discard changes made during the transactional execution.

1 15. (Original) The computer system of claim 14, wherein while discarding
2 changes made during the transactional execution, the execution mechanism is
3 configured to:

4 discard register file changes made during the transactional execution;
5 clear load marks from cache lines;
6 drain store buffer entries generated during transactional execution; and to
7 clear store marks from cache lines.

1 16. (Original) The computer system of claim 13, wherein while
2 terminating the transactional execution, the execution mechanism is additionally
3 configured to branch to a location specified by a corresponding start transactional
4 execution (STE) instruction.

1 17 (Canceled).

1 18. (Original) The computer system of claim 13, wherein while
2 terminating the transactional execution, the execution mechanism is additionally
3 configured to attempt to re-execute the block of instructions.

1 19. (Original) The computer system of claim 13, wherein if the
2 transactional execution of the block of instructions is successfully completes, the
3 execution mechanism is configured to:
4 atomically commit changes made during the transactional execution; and
5 to
6 resume normal non-transactional execution.

1 20. (Original) The computer system of claim 13, wherein the computer
2 system is configured to allow potentially interfering data accesses from other
3 processes to proceed during the transactional execution of the block of
4 instructions.

1 21. (Original) The computer system of claim 13, wherein if an interfering
2 data access from another process is encountered during the transactional
3 execution, the execution mechanism is configured to:
4 discard changes made during the transactional execution; and to
5 attempt to re-execute the block of instructions.

1 22. (Original) The computer system of claim 13, wherein the block of
2 instructions to be executed transactionally comprises a critical section.

1 23. (Original) The computer system of claim 13, wherein the fail
2 instruction is a native machine code instruction of the processor.

1 24. (Original) The computer system of claim 13, wherein the fail
2 instruction is defined in a platform-independent programming language.

1 25. (Currently amended) A computing means that supports that supports a
2 fail instruction to facilitate transactional execution, comprising:
3 a processing means; and
4 an execution means within the processing means;
5 wherein the execution means is configured to execute a start transactional
6 execution instruction to transactionally execute a block of instructions within a
7 program;
8 wherein changes made during the transactional execution are not
9 committed to the architectural state of the processor unless the transactional
10 execution successfully completes; and
11 wherein if the fail instruction is encountered during the transactional
12 execution, the execution means is configured to:
13 terminate the transactional execution without committing
14 results of the transactional execution to the architectural state of
15 the processor, ~~wherein terminating the transactional execution~~
16 ~~involves branching to a location specified by the fail instruction~~
17 retry the transactional execution a specified number of
18 times, and
19 if the fail instruction continues to be encountered, obtain a
20 lock for the block of instructions.